Prime factors are divisors which are prime numbers.
The number itself can also be prime factor.

1) One naive approach could be check all numbers till number if they are divisor and prime
   TC - $O(n \times \sqrt{n})$
   ↑ ↑ Finding sqrt.
   Looping through numbers

2) We can optimize this by finding factors in $\sqrt{n}$ only going till $\sqrt{n}$ & checking if i & n/i are prime numbers & factors.
   TC - For finding factors it'll be $\sqrt{n}$. However, we will have prime check only for factors so it's unpredictable and depends on input.

3) Another more optimized approach would be to loop from 2 to n and if n is divisible by i add i to the list of answer. Also, while n is divisible by 'i' keep dividing it by 'i' till it's no longer divisible by it to handle all multiples of that number. Finally if number is not 1 add it to list as well.

```
public static List<Integer> generatePrimeFactors(int n) {
    List<Integer> primeFactors = new ArrayList<>();
    for (int i = 2; i * i < n; i++) {
        if (n % i == 0)
            primeFactors.add(i);
        while (n % i == 0) { n = n/ i; }
    }
    if (n != 1)
        primeFactors.add(n);
    return primeFactors;
}
```

TC - $O(\sqrt{n} \times \log n)$
↑
This comes from while loop where we are dividing n.
The base for this will keep changing.